Extension of Fill's perfect rejection sampling algorithm to general chains

James Allen Fill * and **Motoya Machida***
Department of Mathematical Sciences
The Johns Hopkins University
Baltimore, MD 21218–2682
U. S. A.
jimfill@jhu.edu
machida@mts.jhu.edu

Duncan J. Murdoch†
Department of Statistical and Actuarial Sciences
University of Western Ontario
London, Ontario N6G 2E9
Canada
murdoch@fisher.stats.uwo.ca

Jeffrey S. Rosenthal†
Department of Statistics
University of Toronto
Toronto, Ontario M5S 3G3
Canada
jeff@math.toronto.edu

April, 1999; last revised July 20, 2000

*These authors' research has been supported in part by NSF grants DMS–9626756 and DMS–9803780, and by the Acheson J. Duncan Fund for the Advancement of Research in Statistics.
†These authors' research has been supported in part by NSERC of Canada.

Abstract

By developing and applying a broad framework for rejection sampling using auxiliary randomness, we provide an extension of the perfect sampling algorithm of Fill (1998) to general chains on quite general state spaces, and describe how use of bounding processes can ease computational burden. Along the way, we unearth a simple connection between the Coupling From The Past (CFTP) algorithm originated by Propp and Wilson (1996) and our extension of Fill's algorithm.

Key words and phrases. Fill's algorithm, Markov chain Monte Carlo, perfect sampling, exact sampling, rejection sampling, interruptibility, coupling from the past, read-once coupling from the past, monotone transition rule, realizable monotonicity, stochastic monotonicity, partially ordered set, coalescence, imputation, time reversal, detection process, bounding process, Polish space, standard Borel space, iterated random functions, tours

AMS 2000 subject classifications. Primary: 60J10, 68U20; secondary: 60G40, 62D05, 65C05, 65C10, 65C40.

1 Introduction

Markov chain Monte Carlo (MCMC) methods have become extremely popular for Bayesian inference problems (see e.g. Gelfand and Smith [16], Smith and Roberts [41], Tierney [43], Gilks et al. [17]), and for problems in other areas, such as spatial statistics, statistical physics, and computer science (see e.g. Fill [11] or Propp and Wilson [37] for pointers to the literature) as a way of sampling approximately from a complicated unknown probability distribution π. An MCMC algorithm constructs a Markov chain with one-step transition kernel K and stationary distribution π; if the chain is run long enough, then under reasonably weak conditions (cf. Tierney [43]) it will converge in distribution to π, facilitating approximate sampling.

One difficulty with these methods is that it is difficult to assess convergence to stationarity. This necessitates the use of difficult theoretical analysis (e.g., Meyn and Tweedie [32], Rosenthal [40]) or problematic convergence diagnostics (Cowles and Carlin [5], Brooks, et al. [2]) to draw reliable samples and do proper inference.

An interesting alternative algorithm, called *coupling from the past* (CFTP), was introduced by Propp and Wilson [37] (see also [38] and [39]) and has been studied and used by a number of authors (including Kendall [26], Møller [33], Murdoch and Green [35], Foss and Tweedie [15], Kendall and Thönnes [28], Corcoran and Tweedie [4], Kendall and Møller [27], Green and Murdoch [18], and Murdoch and Rosenthal [36]). By searching backwards in time until paths from all starting states have coalesced, this algorithm uses the Markov kernel K to sample *exactly* from π.

Another method of perfect simulation, for finite-state stochastically monotone chains, was proposed by Fill [11]. Fill's algorithm is a form of rejection sampling. This algorithm was later extended by Møller and Schladitz [34] and Thönnes [42] to non-finite chains, motivated by applications to spatial point processes. Fill's algorithm has the advantage over CFTP of removing the correlation between the length of the run and the returned value, which eliminates bias introduced by an impatient user or a system crash and so is "interruptible". However, it has been used only for stochastically monotone chains, making heavy use of the ordering of state space elements. In his paper, Fill [11] indicated that his algorithm could be suitably modified to allow for the treatment of "anti-monotone" chains and (see his Section 11.2) indeed to generic chains. A valuable background resource on perfect sampling methods is the annotated bibliography maintained by Wilson [44].

The goal of the present paper is to discuss the modifications to Fill's algorithm needed to apply it to generic chains, on general (not necessarily finite) state spaces. Our basic algorithm is presented in Section 2 as Algorithm 2.1. An infinite-time-window version of Algorithm 2.1 (namely, Algorithm 3.1) is presented in Section 3. A simple illustrative example is presented in Section 4, while rigorous mathematical details are presented in Section 6.

In Section 5 we discuss how the computational burden of tracking all of the trajectories in Algorithm 2.1 can be eased by the use of coalescence detection events in general and bounding processes in particular; these processes take on a very simple form (see Section 7.1) when the state space is partially ordered and the transition rule employed is monotone. A weaker form of monotonicity is also handled in Section 7.

In Section 8 we compare Algorithm 2.1 and CFTP. We also present a simple connection between CFTP and Algorithm 3.1. Finally, in Section 9 we discuss the perfect generation of samples from π of size larger than 1.

We hope that our extension of Fill's algorithm will stimulate further research into this less-used alternative for perfect MCMC simulation.

Notational note: Throughout the paper, we adopt the probabilist's usual shorthand of writing $\{X \in B\}$ for the event $\{\omega \in \Omega : X(\omega) \in B\}$ when X is a random element defined on a sample space Ω.

2 The algorithm in brief

We assume here that our Markov chain may be written in the *stochastic recursive sequence* form

$$\mathbf{X}_s = \phi(\mathbf{X}_{s-1}, \mathbf{U}_s) \tag{2.1}$$

where $(\mathbf{U}_s)$ is an i.i.d. sequence having distribution (say) μ.

Omitting technical details, our interruptible algorithm for generic chains is conceptually quite simple and proceeds as follows.

Algorithm 2.1 Choose and fix a positive integer t, choose an initial state $\mathbf{X}_t$, and perform the following routine. Run the time-reversed chain $\widetilde{K}$ for t steps [see (6.5) for the formal definition of $\widetilde{K}$], obtaining $\mathbf{X}_{t-1}, \ldots, \mathbf{X}_0$ in succession. Then (conditionally given $\mathbf{X}_0, \ldots, \mathbf{X}_t$) generate $\mathbf{U}_1, \ldots, \mathbf{U}_t$ independently, with $\mathbf{U}_s$ chosen from its conditional distribution given (2.1) ($s = 1, \ldots, t$). Then, for each element x of the state space $\mathcal{X}$, compute chains $(\mathbf{Y}_0(x), \ldots, \mathbf{Y}_t(x))$, with $\mathbf{Y}_0(x) := x$ and $\mathbf{Y}_s(x) := \phi(\mathbf{Y}_{s-1}(x), \mathbf{U}_s)$ for $s = 1, 2, \ldots, t$. Note that $\mathbf{Y}_s(\mathbf{X}_0) = \mathbf{X}_s$ for $s = 1, \ldots, t$. Finally, check whether all the values $\mathbf{Y}_t(x)$, $x \in \mathcal{X}$, agree (in which case, of course, they all equal $\mathbf{X}_t$). If they do, we call this *coalescence*, and the routine succeeds and reports $\mathbf{W} := \mathbf{X}_0$ as an observation from π. If not, then the routine fails; we then start the routine again with an independent simulation (perhaps with a fresh choice of t and $\mathbf{X}_t$), and repeat until the algorithm succeeds.

Remark 2.2 The algorithm works for π-almost every deterministic choice of initial state $\mathbf{X}_t$. Alternatively, the algorithm works provided one chooses $\mathbf{X}_t$ from any distribution absolutely continuous with respect to π. See also Remark 6.9(c).

Here is the basic idea of why the algorithm works correctly. Imagine (1) *starting* the construction with $\mathbf{X}_0 \sim \pi$ and independently (2) simulating $\mathbf{U}_1, \ldots, \mathbf{U}_t$. Determination of coalescence and the value of the coalesced paths at time t each rely only on the second piece of randomness. It follows that, conditionally given coalescence, $\mathbf{X}_0$ and $\mathbf{X}_t$ are independent. Hence, conditionally given coalescence and $\mathbf{X}_t$, we will still have $\mathbf{X}_0 \sim \pi$, as desired. The algorithm constructs the random variables in a different order, but conditional on coalescence and the value of $\mathbf{X}_t$, the joint distributions are the same.

Remark 2.3 (a) Note that no assumption is made in Algorithm 2.1 concerning monotonicity or discreteness of the state space.

(b) This algorithm is, like Fill's original algorithm [11], a form of rejection sampling (see, e.g., Devroye [6]). This is explained in Section 2 of [14].

(c) We have reversed the direction of time, and the roles of the kernels K and $\widetilde{K}$, compared to Fill [11].

(d) Algorithm 2.1 is interruptible, in the sense of Fill [11].

(e) Fill's original algorithm [11] also incorporated a search for a good value of t by doubling the previous value of t until the first success. For the most part, we shall not address such issues, instead leaving the choice of t entirely up to the user; but see Section 3.

In Section 6, we will carefully discuss the underlying assumptions for Algorithm 2.1 and the details of its implementation, and also establish rigorously that the algorithm works as desired. This will be done by first developing, and then applying, results in a rather more general framework.

3 A modified algorithm which searches for t

Thus far we have been somewhat sketchy about the choice(s) of t in Algorithm 2.1. As discussed in Remark 2.3(e), one possibility is to run the repetitions of the basic routine independently, doubling t at each stage. However, another possibility is to continue back in time, reusing the already imputed values $\mathbf{U}_s$ and checking again for coalescence. (There is an oblique reference to this alternative in Remark 9.3 of Fill [11].) This idea leads to the following algorithm.

Algorithm 3.1 Choose an initial state $\mathbf{X}_0 \sim \hat{\pi}$, where $\hat{\pi}$ is absolutely continuous with respect to π. Run the time-reversed chain $\widetilde{K}$, obtaining $\mathbf{X}_0, \mathbf{X}_{-1}, \ldots$ in succession. Then (conditionally given $\mathbf{X}_0, \ldots, \mathbf{X}_t$) generate $\mathbf{U}_0, \mathbf{U}_{-1}, \ldots$ independently, with $\mathbf{U}_s$ chosen from its conditional distribution given (2.1) ($s = 0, -1, \ldots$). For $t = 0, 1, \ldots$ and $x \in \mathcal{X}$, set $\mathbf{Y}_{-t}^{(-t)}(x) := x$ and, inductively,

$$\mathbf{Y}_s^{(-t)}(x) := \phi(\mathbf{Y}_{s-1}^{(-t)}(x), \mathbf{U}_s), \quad -t+1 \le s \le 0.$$

If $\mathbf{T} < \infty$ is the smallest t such that

$$\mathbf{Y}_0^{(-t)}(x),\ x \in \mathcal{X},\ \text{all agree} \qquad (\text{and hence all equal } \mathbf{X}_0), \tag{3.1}$$

then the algorithm succeeds and reports $\mathbf{W} := \mathbf{X}_{-\mathbf{T}}$ as an observation from π. If there is no such $\mathbf{T}$, then the algorithm fails.

Here is the basic idea of why the algorithm works correctly. Imagine (1) starting the construction with $\mathbf{W} \sim \pi$, and, independently, (2) simulating $\mathbf{U}_0, \mathbf{U}_{-1}, \ldots$ [and then, after determining $\mathbf{T}$, setting $\mathbf{X}_{-\mathbf{T}} := \mathbf{W}$ and $\mathbf{X}_s := \phi(\mathbf{X}_{s-1}, \mathbf{U}_s)$ for $s = -\mathbf{T}+1, \ldots, 0$]. Determination of $\mathbf{T}$ and the value of $\mathbf{X}_0$ each rely only on the second piece of randomness. It follows that, conditionally given coalescence, $\mathbf{X}_{-\mathbf{T}}$ and $\mathbf{X}_0$ are independent. Hence, conditionally given coalescence and $\mathbf{X}_0$, we will still have $\mathbf{X}_{-\mathbf{T}} \sim \pi$, as desired. As before, the algorithm constructs the random variables in a different order, but conditional on coalescence and the value of $\mathbf{X}_t$, the joint distributions are the same.

Remark 3.2 (a) We need only generate $\mathbf{X}_0, \mathbf{X}_{-1}, \ldots, \mathbf{X}_{-t}$ and then impute $\mathbf{U}_0$, $\mathbf{U}_{-1}, \ldots, \mathbf{U}_{-t+1}$ in order to check whether or not (3.1) holds. Thus, as long as $\mathbf{T} < \infty$, the algorithm runs in finite time.

(b) One can formulate the algorithm rigorously in the fashion of Section 6.2, and verify that it works properly. We omit the details.

(c) Algorithm 3.1 is also interruptible: specifically, $\mathbf{T}$ and $\mathbf{W}$ are conditionally independent given success.

(d) See also the discussion of a "doubling" search strategy in Section 5 below.

4 A simple illustrative example

We illustrate Algorithm 2.1 for a very simple example (for which direct sampling from π would be elementary, of course) and two different choices of transition rule. Consider the discrete state space $\mathcal{X} = \{0, 1, 2\}$, and let π be uniform on $\mathcal{X}$. Let K correspond to simple symmetric random walk with holding probability $1/2$ at the endpoints; that is, putting $k(x, y) := K(x, \{y\})$,

$$\begin{aligned} &k(0,0) = k(0,1) = k(1,0) = k(1,2) = k(2,1) = k(2,2) = 1/2, \\ &k(0,2) = k(1,1) = k(2,0) = 0. \end{aligned}$$

The stationary distribution is π. As for any ergodic birth-and-death chain, K is reversible with respect to π, i.e., $\widetilde{K} = K$. Before starting the algorithm, choose a transition rule; this is discussed further below.

For utter simplicity of description, we choose $t = 2$ and (deterministically) $\mathbf{X}_t = 0$ (say); as discussed near the end of Section 6.2, a deterministic start is

permissible here. We then choose $\mathbf{X}_1 \sim K(0, \cdot)$ and $\mathbf{X}_0 \,|\, \mathbf{X}_1 \sim K(\mathbf{X}_1, \cdot)$. How we proceed from this juncture depends on what we chose (in advance) for ϕ.

One choice is the independent-transitions rule [discussed further in Remarks 6.7(c) and 6.11(b) below]. The algorithm's routine can then be run using 6 independent random bits: these decide $\mathbf{X}_1$ (given $\mathbf{X}_2$), $\mathbf{X}_0$ (given $\mathbf{X}_1$), and the 4 transitions in the second (forward) phase of the routine not already determined from the rule

$$\mathbf{X}_{s-1} \mapsto \mathbf{X}_s \text{ from time } s-1 \text{ to time } s \ (s = 1, 2).$$

There are thus a total of $2^6 = 64$ possible overall simulation results, each having probability 1/64. We check that exactly 12 of these produce coalescence. Of these 12 accepted results, exactly 4 have $\mathbf{X}_0 = 0$, another 4 have $\mathbf{X}_0 = 1$, and a final 4 have $\mathbf{X}_0 = 2$. Thus $\mathbf{P}(\mathbf{C}) = 12/64 = 3/16$, and we confirm that $\mathcal{L}(\mathbf{X}_0|C) = \pi$, so the algorithm is working correctly. (An identical result holds if we had instead chosen $\mathbf{X}_t = 1$ or $\mathbf{X}_t = 2$.)

An alternative choice adapts Remarks 6.7(b) and 6.11(c) to the discrete setting of our present example. We set

$$\phi(\cdot, u) = \begin{cases} \text{the mapping taking } \ 0,1,2 \text{ to } \ 0,0,1, & \text{respectively, if } u = 0 \\ \text{the mapping taking } \ 0,1,2 \text{ to } \ 1,2,2, & \text{respectively, if } u = 1 \end{cases}$$

where u is uniform on $\{0, 1\}$. Choosing $t = 2$ and $\mathbf{X}_t = 0$ as before, the algorithm can now be run with just 2 random bits. In this case we check that exactly 3 of the 4 possible simulation results produce coalescence, 1 each yielding $\mathbf{X}_0 = 0, 1, 2$. Note that $\mathbf{P}(\mathbf{C}) = 3/4$ is much larger for this choice of ϕ. In fact, since ϕ is a monotone transition rule [see Definition 4.2 in Fill [11] or (7.1) below], for the choice $\mathbf{X}_t = 0$ it gives the highest possible value of $\mathbf{P}(\mathbf{C})$ among all choices of ϕ: see Remark 9.3(e) in Fill [11]. It also is a best choice when $\mathbf{X}_t = 2$. [On a minor negative note, we observe that $\mathbf{P}(\mathbf{C}) = 0$ for the choice $\mathbf{X}_t = 1$. Also note that the π-average of the acceptance probabilities $(3/4, 0, 3/4)$, namely, $1/2$, is the probability that forward coupling (or CFTP) done with the same transition rule gives coalescence within 2 time units; this corroborates Remark 6.9(c) below.]

Remark 4.1 Both choices of ϕ are easily extended to handle simple symmetric random walk on $\{0, \ldots, n\}$ for any n. If $\mathbf{X}_t = 0$, then the second (monotone) choice is again best possible. For fixed $c \in (0, \infty)$ and large n, results in Fill [11] and Section 4 of Diaconis and Fill [7] imply that, for $t = cn^2$ and $\mathbf{X}_t = 0$, the routine's success probability is approximately $p(c)$; here $p(c)$ increases smoothly from 0 to 1 as c increases from 0 to ∞. We have not attempted the corresponding asymptotic analysis for the independent-transitions rule.

5 Conservative detection of coalescence

(*Note:* In order to focus on the main ideas, in this Section 5—as in the previous sections—we will suppress measure-theoretic and other technical details. In an earlier draft we provided these details; we trust that the interested reader will be able to do the same, using the rigorous treatment of Algorithm 2.1 in Section 6 as a guide. Also note that the terminology used in this section—detection process, bounding process, etc.—has varied somewhat in the perfect sampling literature.)

Even for large finite state spaces $\mathcal{X}$, determining exactly whether or not coalescence occurs in Algorithm 2.1 can be prohibitively expensive computationally; indeed, in principle this requires tracking each of the trajectories $\vec{\mathbf{Y}}(x) := (\mathbf{Y}_0(x), \ldots, \mathbf{Y}_t(x))$, $x \in \mathcal{X}$, to completion.

However, suppose that $\mathbf{E}$ is an event which is a *subset* of the coalescence event. That is, whenever $\mathbf{E}$ occurs, coalescence occurs (but perhaps not conversely). Assume further that $\mathbf{E}$, like the coalescence event, is an event whose occurrence (or not) is determined solely by $\mathbf{U}_1, \ldots, \mathbf{U}_t$. Then Algorithm 2.1 remains valid if, instead of accepting $\mathbf{W}$ whenever coalescence occurs, we accept only when the event $\mathbf{E}$ occurs. Indeed, the explanation for why Algorithm 2.1 works goes through without change in this case.

It follows that, when implementing Algorithm 2.1, it is permissible to use *conservative detection of coalescence*, i.e., to accept $\mathbf{W}$ as an observation from π if and only if some event $\mathbf{E}$ occurs, provided that $\mathbf{E}$ is a subset of the coalescence event and is a function of $\mathbf{U}_1, \ldots, \mathbf{U}_t$ only. We call such an event $\mathbf{E}$ a *coalescence detection event*.

Similar considerations apply to Algorithm 3.1. Indeed, let $\mathbf{T}$ be as in that algorithm. Let $\mathbf{T}'$ be any other positive-integer-valued random variable, which is completely determined by $\mathbf{U}_0, \mathbf{U}_{-1}, \ldots$ and is such that $\mathbf{T}' \geq \mathbf{T}$. Then Algorithm 3.1 remains valid if we replace $\mathbf{T}$ by $\mathbf{T}'$, i.e., if we report $\mathbf{W}' := \mathbf{X}_{-\mathbf{T}'}$ instead of reporting $\mathbf{X}_{-\mathbf{T}}$. Indeed, the explanation for why Algorithm 3.1 works goes through without change in this case.

For example, we might choose in Algorithm 3.1 to let $\mathbf{T}'$ be the smallest t which is a power of 2 such that (3.1) holds and report $\mathbf{X}_{-\mathbf{T}'}$ instead. This has the computational efficiency advantage (similar to that of CFTP) that we can use a "doubling" search strategy, trying $t = 1, 2, 4, 8, \ldots$ in succession, until we find the first time $t\,(= \mathbf{T}')$ such that (3.1) holds.

In such a case, and in other cases discussed below, conservative coalescence detection will often lead to easier and faster application of our algorithms. Thus, such detection may be of considerable help in practical applications.

5.1 Detection processes

In practice, a coalescence detection event is constructed in terms of a *detection process*. What we mean by this is a stochastic process $\vec{\mathbf{D}} = (\mathbf{D}_0, \ldots, \mathbf{D}_t)$, defined on the same probability space as $\vec{\mathbf{U}} = (\mathbf{U}_1, \ldots, \mathbf{U}_t)$ and $\vec{\mathbf{X}} = (\mathbf{X}_0, \ldots, \mathbf{X}_t)$, together with a subset Δ of its state space $\mathcal{D}$, such that

(a) $\vec{\mathbf{D}}$ is constructed solely from $\vec{\mathbf{U}}$, and

(b) $\{\mathbf{D}_s \in \Delta \text{ for some } s \le t\} \subseteq \{\mathbf{Y}_t(x) \text{ does not depend on } x\}$.

Then $\mathbf{E} := \{\mathbf{D}_s \in \Delta \text{ for some } s \le t\}$ is a coalescence detection event.

Remark 5.1 In practice, $\vec{\mathbf{D}}$ usually evolves Markovianly using $\vec{\mathbf{U}}$; more precisely, it is typically the case that there exists deterministic $d_0 \in \mathcal{D}$ such that $\mathbf{D}_0 = d_0$ and $\vec{\mathbf{D}}$ has the stochastic recursive sequence form [paralleling (2.1)]

$$\mathbf{D}_s := \delta(\mathbf{D}_{s-1}, \mathbf{U}_s), \quad 1 \le s \le t.$$

The important consequence is that, having determined the trajectory $\vec{\mathbf{X}}$ and the imputed $\vec{\mathbf{U}}$, the user need only follow a single trajectory in the forward phase of the routine, namely, that of $\vec{\mathbf{D}}$.

Example 5.2 We sketch two illustrative examples of the use of detection processes that do not immediately fall (see Remark 5.3 below) into the more specific settings of Sections 5.2 or 7.1. We hasten to point out, however, that because of the highly special structure of these two examples, efficient implementation of Algorithm 2.1 avoids the use of the forward phase altogether; this is discussed for example (a) in Fill [12].

(a) Our first example is provided by the move-to-front (MTF) rule studied in [12]. Let K be the Markov kernel corresponding to MTF with independent and identically distributed record requests corresponding to probability weight vector $(w_1, \ldots, w_n)$; see (2.1) of [12] for specifics. The arguments of Section 4 of [12] show that if $\mathbf{D}_s$ is taken to be the set of all records requested at least once among the first s requests and Δ is taken to consist of all $(n-1)$-element subsets of the records $1, \ldots, n$, then $\vec{\mathbf{D}}$ is a detection process. Similar detection processes can be built for the following generalizations of MTF: move-to-root for binary search trees (see Dobrow and Fill [9] [10]) and MTF-like shuffles of hyperplane arrangement chambers and more general structures (see Bidigare, et al. [1] and Brown and Diaconis [3]).

(b) A second example of quite similar spirit is provided by the (now well-known) Markov chain $(\mathbf{X}_t)$ for generating a random spanning arborescence of the underlying weighted directed graph, with vertex set $\mathcal{U}$, of a Markov chain $(\mathbf{U}_t)$ with state space $\mathcal{U}$ and kernel q. Consult Propp and Wilson [39] (who also discuss

a more efficient "cycle-popping" algorithm) for details. We consider here only the special case that $(\mathbf{U}_t)$ is an i.i.d. sequence, i.e., that $q(v, w) \equiv q(w)$. A transition rule ϕ for the chain $(\mathbf{X}_t)$ is created as follows: for vertex u and arborescence x with root r, $\phi(x, u)$ is the arborescence obtained from x by adding an arc from r to u and deleting the unique arc in x whose tail is u. Then it can be shown that if $\mathbf{D}_s$ is taken to be the set of all vertices appearing at least once in $(\mathbf{U}_1, \ldots, \mathbf{U}_s)$ and $\Delta := \{\mathcal{U}\}$, then $\vec{\mathbf{D}}$ is a detection process.

5.2 Bounding processes

We obtain a natural and useful example of a detection process $\vec{\mathbf{D}}$ when (a) $\vec{\mathbf{D}}$ is constructed solely from $\vec{\mathbf{U}}$, (b) the corresponding state space $\mathcal{D}$ is some collection of subsets of $\mathcal{X}$, with

$$\Delta := \{\{z\} : \ z \in \mathcal{X}\},$$

and

$$\text{(c)} \quad \mathbf{D}_s \supseteq \{\mathbf{Y}_s(x) : \ x \in \mathcal{X}\}.$$

The concept is simple: in this case, each set $\mathbf{D}_s$ is just a "conservative estimate" (i.e., a superset) of the corresponding set $\{\mathbf{Y}_s(x) : x \in \mathcal{X}\}$ of trajectory values; thus if $\mathbf{D}_s = \{z\}$, then the trajectories $\vec{\mathbf{Y}}(x)$ are coalesced to state z at time s and remain coalesced thereafter. We follow the natural impulse to call such a set-valued detection process a *bounding process*. Such bounding processes arise naturally in the contexts of monotone (and anti-monotone) transition rules, and have been used by many authors: see Section 7.1. Other examples of bounding processes can be found in Häggström and Nelander [20] (for CFTP) and in works of Huber: see [22] and [23] in connection with CFTP and [24] in connection with our algorithm.

Of course, nothing is gained, in comparison to tracking all the trajectories, by the use of a bounding process unless the states of $\mathcal{D}$ have more concise representations than those of generic subsets of $\mathcal{X}$; after all, we could always choose $\mathcal{D} = 2^{\mathcal{X}}$ and $\mathbf{D}_s = \{\mathbf{Y}_s(x) : x \in \mathcal{X}\}$. One rather general, and frequently applied, setting where compact representations are possible, discussed in Section 7.1, is that of a realizably monotone chain on a partially ordered set (poset) $\mathcal{X}$. See the references listed in Remark 7.4(a) for examples of the use of bounding processes for such chains.

See also Remark 7.4(b) for a brief discussion of the closely related concept of "dominating process."

Remark 5.3 With our algorithm, one can take advantage of detection processes that are more general than bounding processes. Consider Example 5.2(a) as an illustrative case. Suppose we start the first (i.e., time-reversed) phase of our algorithm in the identity permutation (as is done in [12]). When we run the forward-time phase of the algorithm, we only need to keep track of the *unordered* set of

records requested; this is what the detection process $\vec{\mathbf{D}}$ of Example 5.2(a) does. If the event $\mathbf{E}$ of Section 5.1 occurs, that is, if all, or all but one, of the records have been requested at least once by time t, then we have detected coalescence. Then, without further work, we know the state to which the trajectories have coalesced: namely, our initial identity permutation. To maintain a bounding process for this same example, we would have to do more work: we would have to keep track of the *ordered* set of records requested, ordered by most recent request. (Note that a bounding process *could* be constructed by combining information from the dominating process and the trajectory generated in the algorithm's initial backward phase. But it is not useful to do so.)

6 A mathematically rigorous framework

6.1 The formal framework

We now formally set up a general framework for rejection sampling using auxiliary randomness, paying careful attention to technical details. We will apply this framework not only to provide a rigorous treatment of Algorithm 2.1 (in Section 6.2), but also (independently) to handle a variant (Algorithm 7.2) that applies to a stochastically monotone kernel K.

We need to set up two probability spaces. In our main application to Algorithm 2.1, the first space $(\Omega, \mathcal{A}, P)$—designated in ordinary typeface—will be useful for theoretical considerations and for the computation of certain conditional probability distributions. The second space $(\mathbf{\Omega}, \mathbf{A}, \mathbf{P})$—designated in boldface type—will be the probability space actually simulated when the algorithm is run. All random variables defined on the first space (respectively, second space) will also be designated in ordinary typeface (resp., boldface type). We have chosen this notational system to aid the reader: Corresponding variables, such as X_0 and $\mathbf{X}_0$, will play analogous roles in the two spaces.

Recall that for a measurable space $(\mathcal{Z}, \mathcal{F})$ and a (not necessarily measurable) subset $E \subseteq \mathcal{Z}$, the *trace σ-field* (on the sample space E) is the σ-field $\{E \cap F : F \in \mathcal{F}\}$. In setting up both probability spaces, we assume:

(i) $(\mathcal{X}, \mathcal{B})$ and $(\mathcal{X}', \mathcal{B}')$ are measurable spaces, $B' \subseteq \mathcal{X}'$, and $\mathcal{B}''$ is the trace σ-field for B'

(ii) π is a probability measure on $\mathcal{B}$.

For our first probability space, we also assume that

(iii) $(\Omega, \mathcal{A}, P)$ is a probability space on which are defined mappings $X : \Omega \to \mathcal{X}$ and $X' : \Omega \to \mathcal{X}'$ and $Y : \Omega \to \mathcal{X}'$ and a set $C \subseteq \Omega$

(iv) X is measurable from $\mathcal{A}$ to $\mathcal{B}$, with $\mathcal{L}(X) = PX^{-1} = \pi$

and that the following two conditions hold, for every $B'' \in \mathcal{B}''$:

(v) $\{X' \in B''\} \cap C \in \mathcal{A}$ and $\{Y \in B''\} \cap C \in \mathcal{A}$

(vi) $P(\{X' \in B''\} \cap C \,|\, X \in B) = P(\{Y \in B''\} \cap C)$ for all $B \in \mathcal{B}$ with $\pi(B) > 0$.

Notice that (v) and (vi) are implied by

(v′) $\{X' \in B''\} \cap C \in \mathcal{A}$, $Y = X'$ on the set C, and X and the event $\{Y \in B''\} \cap C$ are independent.

Proposition 6.1 *Under assumptions* (i)–(vi)*,*

$$P(\{X' \in B''\} \cap C \cap \{X \in B\}) = P(\{X' \in B''\} \cap C)\, \pi(B) \tag{6.1}$$

for every $B \in \mathcal{B}$ and every $B'' \in \mathcal{B}''$.

Proof. We may assume $\pi(B) > 0$, in which case (vi) implies

$$P(\{X' \in B''\} \cap C \cap \{X \in B\}) = P(\{Y \in B''\} \cap C)\, \pi(B), \tag{6.2}$$

$$P(\{X' \in B''\} \cap C) = P(\{Y \in B''\} \cap C) \tag{6.3}$$

for general $B \in \mathcal{B}$ and for $B = \mathcal{X}$, respectively. Substituting (6.3) into (6.2) gives (6.1). ∎

Two corollaries follow immediately:

Corollary 6.2 *Under assumptions* (i)–(vi)*, if $P(\{X' \in B'\} \cap C) > 0$, then*

$$\mathcal{L}(X \,|\, \{X' \in B'\} \cap C) = \pi.$$

Corollary 6.3 *Suppose assumptions* (i)–(vi) *hold for $B' = \mathcal{X}'$; in particular,* (v) *then implies that $C \in \mathcal{A}$. Suppose also that X' is measurable from $\mathcal{A}$ to $\mathcal{B}'$. Assume $P(C) > 0$. Then*

$$X' \text{ and } X \text{ are conditionally independent given } C, \text{ and } \mathcal{L}(X|C) = \pi.$$

Now we set up the second probability space. Specifically, consider the following assumptions: that

(vii) $(\mathbf{\Omega}, \mathbf{A}, \mathbf{P})$ is a probability space on which are defined mappings $\mathbf{X} : \mathbf{\Omega} \to \mathcal{X}$ and $\mathbf{X}' : \mathbf{\Omega} \to \mathcal{X}'$ and a set $\mathbf{C} \subseteq \mathbf{\Omega}$

(viii) $\mathbf{X}$ is measurable from $\mathbf{A}$ to $\mathcal{B}$,

that, for every $B'' \in \mathcal{B}''$, we have

(ix) $\{\mathbf{X}' \in B''\} \cap \mathbf{C} \in \mathbf{A}$,

and that we have the following basic connection between the two spaces:

(x) The measure

$$\mathbf{P}(\{\mathbf{X}' \in dx'\} \cap \mathbf{C} \cap \{\mathbf{X} \in dx\})$$

on the product space $(B' \times \mathcal{X}, \mathcal{B}'' \otimes \mathcal{B})$ has a density $D(x, x') \equiv D(x')$ $(x' \in B')$ that doesn't depend on $x \in \mathcal{X}$ with respect to the measure

$$P(\{X' \in dx'\} \cap\ C \cap \{X \in dx\}).$$

Notice that (x) is implied by conditions (x′)–(x‴), wherein $\mathcal{L}(\mathbf{X}') = \mathbf{P}(\mathbf{X}')^{-1}$:

(x′) $B' = \mathcal{X}'$, and X' and $\mathbf{X}'$ are measurable (from $\mathcal{A}$ and $\mathbf{A}$, respectively, to $\mathcal{B}'$)

(x″) $\mathcal{L}(\mathbf{X}') \ll \mathcal{L}(X')$, with Radon–Nikodym derivative D

(x‴) There exists a conditional subprobability distribution

$$P(C \cap \{X \in dx\} \,|\, X' = x'), \quad x' \in \mathcal{X}',$$

which also serves as conditional subprobability distribution

$$\mathbf{P}(\mathbf{C} \cap \{\mathbf{X} \in dx\} \,|\, \mathbf{X}' = x'), \quad x' \in \mathcal{X}'.$$

It is now key that the results of Proposition 6.1 and Corollaries 6.2 and 6.3 carry over to our second space:

Proposition 6.4 *Under assumptions* (i)–(x)*,*

$$\mathbf{P}(\{\mathbf{X}' \in B''\} \cap \mathbf{C} \cap \{\mathbf{X} \in B\}) = \mathbf{P}(\{\mathbf{X}' \in B''\} \cap \mathbf{C})\, \pi(B) \tag{6.4}$$

for every $B \in \mathcal{B}$ and every $B'' \in \mathcal{B}''$.

Proof. Let D be the Radon–Nikodym derivative guaranteed by assumption (x). Then, using Proposition 6.1,

$$\begin{aligned} \mathbf{P}(\{\mathbf{X}' \in B''\} \cap \mathbf{C} \cap \{\mathbf{X} \in B\}) &= \int_{B''} D(x')\, P(\{X' \in dx'\} \cap C \cap \{X \in B\}) \\ &= \int_{B''} D(x')\, P(\{X' \in dx'\} \cap C)\, \pi(B). \end{aligned}$$

As usual, setting $B = \mathcal{X}$ and substituting, we obtain (6.4). ∎

Corollary 6.5 *Under assumptions* (i)–(x)*, if* $\mathbf{P}(\{\mathbf{X}' \in B'\} \cap \mathbf{C}) > 0$*, then*

$$\mathcal{L}(\mathbf{X} \,|\, \{\mathbf{X}' \in B'\} \cap \mathbf{C}) = \pi.$$

Corollary 6.6 *Suppose assumptions* (i)–(x) *hold for $B' = \mathcal{X}'$. Suppose also that $\mathbf{X}'$ is measurable from $\mathbf{A}$ to $\mathcal{B}'$. Assume $\mathbf{P}(\mathbf{C}) > 0$. Then*

$\mathbf{X}'$ and $\mathbf{X}$ are conditionally independent given $\mathbf{C}$, and $\mathcal{L}(\mathbf{X}|\mathbf{C}) = \pi$.

6.2 Details for Algorithm 2.1

The goal of this subsection is to describe when and how Algorithm 2.1 can be applied legitimately.

The space $(\mathcal{X}, \mathcal{B})$*:* Recall that a *Polish space* is a complete separable metric space. For convenience, we shall assume that the measurable state space $(\mathcal{X}, \mathcal{B})$ of interest (on which the probability measure π of interest is defined) is isomorphic to a Borel subset of a Polish space (with its trace Borel σ-field). This assumption will at once guarantee the existence of such objects as conditional distributions that would otherwise require individual arguments or assumptions. We call such a space a *standard Borel space*. Our assumption should cover most cases of applied interest.

The kernel K *and its time-reversal* $\widetilde{K}$*:* Let $K : \mathcal{X} \times \mathcal{B} \to [0, 1]$ be a Markov transition kernel on $\mathcal{X}$; that is, we suppose that $K(x, \cdot)$ is a probability measure on $\mathcal{B}$ for each $x \in \mathcal{X}$ and that $K(\cdot, B)$ is a $\mathcal{B}$-measurable function for each $B \in \mathcal{B}$. The kernel is chosen (by the user) so that π is a stationary distribution, i.e., so that

$$\int_{\mathcal{X}} \pi(dx) K(x, dy) = \pi(dy) \quad \text{on } \mathcal{X}.$$

Since $(\mathcal{X}, \mathcal{B})$ is standard Borel, there exists "a conditional distribution the other way around"—more precisely, a Markov kernel $\widetilde{K}$ on $\mathcal{X}$ satisfying

$$\pi(dx) K(x, dy) = \pi(dy) \widetilde{K}(y, dx) \quad \text{on } \mathcal{X} \times \mathcal{X}. \tag{6.5}$$

Given π and K, the kernel $\widetilde{K}(y, dx)$ is $\pi(dy)$-almost surely uniquely defined. We choose and fix such a $\widetilde{K}$.

The transition rule ϕ*:* It can be shown that there exists a transition rule which can be used to drive the construction of the Markov chain of interest. More precisely, our assumption that $(\mathcal{X}, \mathcal{B})$ is standard Borel implies that there exists a standard Borel space $(\mathcal{U}, \mathcal{F})$, a product-measurable function $\phi : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$, and a probability measure μ on $\mathcal{F}$, such that

$$K(x, B) = \mu\{u : \phi(x, u) \in B\}, \quad x \in \mathcal{X}, \quad B \in \mathcal{B}. \tag{6.6}$$

Such ϕ (with accompanying μ) is sometimes called a *transition rule*. We choose and fix such a (ϕ, μ).

Remark 6.7 (a) Conversely, if ϕ has the stated properties and K is *defined* by (6.6), then K is a Markov kernel.

(b) A transition rule ϕ can always be found that uses $(\mathcal{U}, \mathcal{F}, \mu) = ([0, 1], \text{Borels}, \text{uniform distribution})$. The proof of existence (cf. Theorem 1.1 in Kifer [29] and Remark (iv) at the end of Section 5.2 in Diaconis and Freedman [8]) makes use of

inverse probability transforms and certain standard reduction arguments. In the special case that $(\mathcal{X}, \mathcal{B}) = ([0,1], \text{Borels})$, we can in fact use

$$\phi(x,u) \equiv G(x,u)$$

where $G(x,\cdot)$ is the usual inverse probability transform corresponding to the distribution function $u \mapsto K(x,[0,u])$.

(c) If $(\mathcal{X}, \mathcal{B})$ is any discrete space (i.e., if $\mathcal{X}$ is countable and $\mathcal{B}$ is the total σ-field), a very simple alternative choice is the following "independent-transitions" transition rule. Let $\mathcal{U} = \mathcal{X}^{\mathcal{X}}$ (with $\mathcal{F}$ the product σ-algebra), let μ be product measure with xth marginal $K(x,\cdot)$ ($x \in \mathcal{X}$), and let ϕ be the evaluation function

$$\phi(x,u) := u(x).$$

(d) Many interesting examples of transition rules can be found in the literature, including Diaconis and Freedman [8] and the references cited in Section 1.

(e) Usually there is a wealth of choices of transition rule, and the art is to find one giving rapid and easily detected coalescence. Without going into details at this point, we remark that the transition rule in (c) usually performs quite badly, while transition rules having a certain monotonicity property will perform well under monotonicity assumptions on K.

The Markov chain and the first probability space: From our previous comments it is now easy to see that there exists a standard Borel space $(\mathcal{U}, \mathcal{F})$, a transition rule (ϕ, μ), and a probability space $(\Omega, \mathcal{A}, P)$ on which are defined independent random variables $X_0, U_1, U_2, \ldots, U_t$ with $X_0 \sim \pi$ and each $U_s \sim \mu$. Now inductively define

$$X_s := \phi(X_{s-1}, U_s), \quad 1 \le s \le t. \tag{6.7}$$

Then $\vec{X} := (X_0, \ldots, X_t)$ is easily seen to be a stationary Markov chain with kernel K, in the sense that

$$P(X_0 \in dx_0, \ldots, X_t \in dx_t) = \pi(dx_0)K(x_0, dx_1)\cdots K(x_{t-1}, dx_t) \quad \text{on } \mathcal{X}^{t+1}. \tag{6.8}$$

In fact, for each $x \in \mathcal{X}$ we obtain a chain with kernel K started from x by defining $Y_0(x) := x$ and, inductively,

$$Y_s(x) := \phi(Y_{s-1}(x), U_s).$$

Let $\vec{Y}(x) := (Y_0(x), \ldots, Y_t(x))$. In this notation we have $\vec{Y}(X_0) = \vec{X}$. Recalling the notational note at the end of Section 1, let

$$C := \{Y_t(x) \text{ does not depend on } x\} \tag{6.9}$$

denote the set of sample points ω for which the trajectories $\vec{Y}(x)$ have all coalesced by time t. We assume that C belongs to the σ-field $\sigma\langle\vec{U}\rangle$ generated by $\vec{U} := (U_1, \ldots, U_t)$.

Remark 6.8 For this remark, suppose that $\mathcal{X}$ *is* a Borel subset of a Polish space (and hence a separable metric space in its own right). We will prove that continuity of the transition rule $\phi(x,u)$ in $x \in \mathcal{X}$ for each $u \in \mathcal{U}$ is sufficient for $C \in \sigma\langle\vec{U}\rangle$, and we note that this is automatic if $\mathcal{X}$ is discrete.

As guaranteed by separability of $\mathcal{X}$, let D be a countable dense subset of $\mathcal{X}$. Given $z \in \mathcal{X}$ and $\epsilon \geq 0$, let $B_z(\epsilon)$ denote the closed ball of radius ϵ centered at z.

Suppose that $\phi(\cdot,u)$ is continuous for each $u \in \mathcal{U}$. Define the iterates $\phi^s : \mathcal{X} \times \mathcal{U}^s \to \mathcal{X}$ $(s = 1,2,\ldots,t)$ inductively by $\phi^1 := \phi$ and

$$\phi^s(x;u_1,\ldots,u_s) := \phi(\phi^{s-1}(x;u_1,\ldots,u_{s-1});u_s).$$

Note that ϕ^t is, like ϕ, continuous in its first argument, and that $Y_t(x) = \phi^t(x;\vec{U})$. Using the separability of $\mathcal{X}$, it is not hard to show that

$$C = \cap_{n=1}^{\infty} \cup_{z\in D} \cap_{x\in D} \{\phi^t(x;\vec{U}) \in B_z(1/n)\},$$

from which the desired measurability of C is evident.

Now observe that conditions (i)–(iv) and (v′) in Section 6.1 are satisfied by fixing $x_0^* \in \mathcal{X}$ arbitrarily and taking

$$\begin{aligned} (\mathcal{X}',\mathcal{B}') &= (\mathcal{X},\mathcal{B}), \quad B' = \mathcal{X}', \quad \mathcal{B}'' = \mathcal{B}, \\ X &= X_0, \quad X' = X_t, \quad Y = Y_t(x_0^*), \quad C \text{ as at (6.9)}. \end{aligned} \tag{6.10}$$

Note that the independence in (v′) follows from the fact that X_0 and $\vec{U}$ have been chosen to be independent.

The second probability space and the algorithm: The key to setting up the second probability space is to satisfy assumption (x‴) in Section 6.1. In calculating the first-space conditional distribution mentioned there, we will make use of the auxiliary randomness provided by $X_1,\ldots,X_{t-1}$ and $\vec{U}$ and compute in stages. First observe from (6.8) and repeated use of (6.5) that

$$P(X_0 \in dx_0,\ldots,X_{t-1} \in dx_{t-1} \,|\, X_t = x_t) = \widetilde{K}(x_t,dx_{t-1}) \cdots \widetilde{K}(x_1,dx_0)$$

serves as a conditional distribution for $(X_0,\ldots,X_{t-1})$ given $X_t = x_t$. Next, we will discuss in Section 6.3 how to compute $\mathcal{L}(\vec{U} \,|\, \vec{X} = \vec{x})$. Finally, from our assumption that $C \in \sigma(\vec{U})$, it follows that we can write the indicator $Z := I_C$ as $Z = \Gamma(\vec{U})$ for some product-measurable $\Gamma : \mathcal{U}^t \to \{0,1\}$, and one can check the intuitively obvious assertion that unit mass at $\Gamma(\vec{u})$ serves as a conditional distribution for Z given $(\vec{X},\vec{U}) = (\vec{x},\vec{u})$. We get the conditional distribution in (x‴) by chaining together the conditional distributions we have computed and integrating out the auxiliary variables, in the obvious and standard fashion.

Moreover, our discussion has indicated how to set up and *simulate* the second space. To satisfy assumption (x″) in Section 6.1, we assume that the law of $\mathbf{X}_t$ chosen by the user is absolutely continuous with respect to π; of course we do *not* assume that the user can compute the Radon–Nikodym derivative D. [For example, in the common situation that $(\mathcal{X}, \mathcal{B})$ is discrete and $\pi(x) > 0$ for every $x \in \mathcal{X}$, the value of $\mathbf{X}_t$ can be chosen deterministically and arbitrarily.] Having chosen $\mathbf{X}_t = x_t$, the user draws an observation $\mathbf{X}_{t-1} = x_{t-1}$ from $\widetilde{K}(x_t, \cdot)$, then an observation $\mathbf{X}_{t-2} = x_{t-2}$ from $\widetilde{K}(x_{t-1}, \cdot)$, etc. Next, having chosen $\vec{\mathbf{X}} = \vec{x}$ [i.e., $(\mathbf{X}_0, \ldots, \mathbf{X}_t) = (x_0, \ldots, x_t)$], the user draws an observation $\vec{\mathbf{U}} = \vec{u}$ from $\mathcal{L}(\vec{U} \,|\, \vec{X} = \vec{x})$. Finally, the user sets $\mathbf{Z} = \Gamma(\vec{u})$ and declares that $\mathbf{C}$, or *coalescence*, has occurred if and only if $\mathbf{Z} = 1$. With the definitions (6.10), $\mathbf{C}$ as above, and

$$\mathbf{X} = \mathbf{X}_0, \quad \mathbf{X}' = \mathbf{X}_t,$$

assumptions (i)–(x) are routinely verified. According to Corollary 6.6, if $\mathbf{P}(\mathbf{C}) > 0$, then $\mathcal{L}(\mathbf{X}_0|\mathbf{C}) = \pi$. It follows that the conditional distribution of output from Algorithm 2.1 given that it ultimately succeeds (perhaps only after many iterations of the basic routine) is π, as desired.

Remark 6.9 (a) If $\mathbf{P}(\mathbf{C}) > 0$ for suitably large t, then ultimate success is (a.s.) guaranteed if the successive choices of t become large. A necessary condition for ultimate positivity of $\mathbf{P}(\mathbf{C})$ is uniform ergodicity of K. This condition is also sufficient, in the (rather weak) sense that if K is uniformly ergodic, then there exists a finite integer m and a transition rule ϕ_m for the m-step kernel K^m such that Algorithm 2.1, applied using ϕ_m, has $\mathbf{P}(\mathbf{C}) > 0$ when t is chosen sufficiently large. Compare the analogous Theorem 4.2 for CFTP in Foss and Tweedie [15].

A similar remark applies to Algorithm 3.1.

(b) Just as discussed in Fill [11] (see especially the end of Section 7 there), the algorithm (including its repetition of the basic routine) we have described is interruptible; that is, its running time (as measured by number of Markov chain steps) and output are independent random variables, conditionally given that the algorithm eventually terminates.

(c) If the user chooses the value of $\mathbf{X}_t$ ($= z$, say) deterministically, then all that can be said in general is that the algorithm works properly for π-a.e. such choice. In this case, let the notation $\mathbf{P}_z(\mathbf{C})$ reflect the dependence of $\mathbf{P}(\mathbf{C})$ on the initial state z. Then clearly

$$\int \mathbf{P}_z(\mathbf{C})\, \pi(dz) = P(C),$$

which is the unconditional probability of coalescence in our first probability space and therefore equal to the probability that CFTP terminates over an interval of width t. This provides a first link between CFTP and Algorithm 2.1. Very roughly

recast, the distribution of running time for CFTP is the stationary mixture, over initial states, of the distributions of running time for Algorithm 2.1. For further elaboration of the connection between the two algorithms, see Section 8.2.

6.3 Imputation

In order to be able to run Algorithm 2.1, the user needs to be able to impute $\vec{U}$ from $\vec{X}$, i.e., to draw from $\mathcal{L}(\vec{U} \,|\, \vec{X} = \vec{x})$. In this subsection we explain how to do this.

We proceed heuristically at first:

$$
\begin{aligned}
& P(\vec{U} \in d\vec{u} \,|\, \vec{X} = \vec{x}) \\
&\quad = \; P(\vec{U} \in d\vec{u} \,|\, X_0 = x_0,\, \phi(x_0, U_1) = x_1,\, \ldots,\, \phi(x_{t-1}, U_t) = x_t) \quad \text{by (6.7)} \\
&\quad = \; P(\vec{U} \in d\vec{u} \,|\, \phi(x_0, U_1) = x_1,\, \ldots,\, \phi(x_{t-1}, U_t) = x_t) \quad \text{by indep. of } X_0 \text{ and } \vec{U} \\
&\quad = \; P(U_1 \in du_1 \,|\, \phi(x_0, U_1) = x_1) \times \cdots \times P(U_t \in du_t \,|\, \phi(x_{t-1}, U_t) = x_t) \\
&\qquad\qquad \text{by independence of } U_1, \ldots, U_t \\
&\quad = \; P(U_1 \in du_1 \,|\, \phi(x_0, U_1) = x_1) \times \cdots \times P(U_1 \in du_t \,|\, \phi(x_{t-1}, U_1) = x_t) \\
&\qquad\qquad \text{since } U_1, \ldots, U_t \text{ are identically distributed} \\
&\quad = \; P(U_1 \in du_1 \,|\, X_0 = x_0,\, X_1 = x_1) \times \cdots \times P(U_1 \in du_t \,|\, X_0 = x_{t-1},\, X_1 = x_t),
\end{aligned}
$$

where the last equality is justified in the same fashion as for the first two.

In fact, the result of this heuristic calculation is rigorously correct; its proof is an elementary but not-entirely-trivial exercise in the use of conditional probability distributions. The existence (and a.s. uniqueness) of a conditional distribution $\mathcal{L}(U_1 \,|X_0 = \cdot,\, X_1 = \cdot)$ is guaranteed by the fact that $(\mathcal{U}, \mathcal{F})$ is standard Borel; moreover,

Lemma 6.10 *The t-fold product of the measures*

$$
P(U_1 \in du_1 \,|\, X_0 = x_0,\, X_1 = x_1),\; \ldots,\; P(U_1 \in du_t \,|\, X_0 = x_{t-1},\, X_1 = x_t)
$$

serves as a conditional probability distribution $P(\vec{U} \in d\vec{u} \,|\, \vec{X} = \vec{x})$.

In setting up the second probability space, therefore, the user, having chosen $\vec{\mathbf{X}} = \vec{x}$, draws an observation $\vec{\mathbf{U}} = \vec{u}$ by drawing $\mathbf{U}_1, \ldots, \mathbf{U}_t$ independently, with $\mathbf{U}_s$ chosen according to the distribution $\mathcal{L}(U_1 \,|\, X_0 = x_{s-1},\, X_1 = x_s)$.

Remark 6.11 (a) There are subtleties involved in the rigorous proof of Lemma 6.10. In particular, there is no justification apparent to us, in general, that the conditional distributions $\mathcal{L}(U_1 \,|\, \phi(x_0, U_1) = \cdot)$, one for each fixed $x_0 \in \mathcal{X}$, can be chosen in such a way that $P(U_1 \in F \,|\, \phi(x_0, U_1) = x_1)$ is jointly measurable in (x_0, x_1) for

each $F \in \mathcal{F}$. Nevertheless, if we *assume* such measurability, then one can show rigorously that

$$P(U_1 \in du_1 \,|\, \phi(x_0, U_1) = x_1) \times \cdots \times P(U_1 \in du_t \,|\, \phi(x_{t-1}, U_1) = x_t)$$

serves as $P(\vec{U} \in d\vec{u} \,|\, \vec{X} = \vec{x})$.

(b) If $(\mathcal{X}, \mathcal{B})$ is discrete, then of course the measurability in (a) is automatic. Suppose we use the "independent-transitions" rule ϕ discussed in Remark 6.7(c). Then the measure μ, but with the x_0th marginal replaced by δ_{x_1}, serves as $\mathcal{L}(U_1 \,|\, \phi(x_0, U_1) = x_1) = \mathcal{L}(U_1 \,|\, U_1(x_0) = x_1)$ and therefore as $\mathcal{L}(U_1 \,|\, X_0 = x_0,\, X_1 = x_1)$. Informally stated, having chosen $\mathbf{X}_s = x_s$ and $\mathbf{X}_{s-1} = x_{s-1}$, the user imputes the forward-trajectory transitions from time $s-1$ to time s in Algorithm 2.1 by declaring that the transition from state x_{s-1} is to state x_s and that the transitions from other states are chosen independently according to their usual non-$\vec{X}$-conditioned distributions.

(c) As another example, suppose that $\mathcal{X} = [0, 1]$ and we use the inverse probability transform transition rule discussed in Remark 6.7(b). Suppose also that each distribution function $F(x_0, \cdot) = K(x_0, [0, \cdot])$ is strictly increasing and onto $[0, 1]$ and that $F(x_0, x_1)$ is jointly Borel-measurable in x_0 and x_1. Then $\delta_{F(x_0,x_1)}$ serves as $\mathcal{L}(U_1 \,|\, X_0 = x_0,\, X_1 = x_1)$. Informally stated, a generated pair $(\mathbf{X}_s, \mathbf{X}_{s-1}) = (x_s, x_{s-1})$ completely determines the value $F(x_{s-1}, x_s)$ for $\mathbf{U}_s$.

7 Monotonicity

Throughout Section 7 we suppose that $(\mathcal{X}, \mathcal{B})$ is a Polish space with (Borel σ-field and) closed partial order $\leq$; the meaning of *closed* here is that $\{(x, y) \in \mathcal{X} \times \mathcal{X} : x \leq y\}$ is assumed to be closed in the product topology. [For example, closedness is automatic for any partial order if $(\mathcal{X}, \mathcal{B})$ is discrete.] We also assume that there exist (necessarily unique) elements $\hat{0}$ and $\hat{1}$ in $\mathcal{X}$ (called *bottom element* and *top element*, respectively) such that $\hat{0} \leq x \leq \hat{1}$ for all $x \in \mathcal{X}$.

We require the notion of stochastic monotonicity, according to the following definitions (which extend Definition 4.1 of Fill [11] to our more general setting).

Definition 7.1 (a) A subset B of $\mathcal{X}$ is called a *down-set* or *order ideal* if, whenever $x \in B$ and $y \leq x$, we have $y \in B$.

(b) Given two probability measures ν_1 and ν_2 on $\mathcal{B}$, we say that $\nu_1 \leq \nu_2$ *stochastically*, and write $\nu_1 \preceq \nu_2$, if $\nu_1(B) \geq \nu_2(B)$ for every closed down-set B.

(c) A kernel K is said to be *stochastically monotone* (SM) if $K(x, \cdot) \preceq K(y, \cdot)$ whenever $x \leq y$.

The main goal of this section is to describe an analogue of Algorithm 2.1 which applies when only stochastic monotonicity of K is assumed. Here is a rough formulation; the details will be discussed in Section 7.2.

Algorithm 7.2 Consider a stochastically monotone kernel K on a partially ordered set with bottom element $\hat{0}$ and top element $\hat{1}$. Choose and fix a positive integer t, set $\mathbf{X}_t = \hat{0}$, and perform the following routine. Run the time-reversed chain $\widetilde{K}$ for t steps, obtaining $\mathbf{X}_t, \mathbf{X}_{t-1}, \ldots, \mathbf{X}_0$ in succession. Then, reversing the direction of time, generate a chain, say $\vec{\mathbf{Y}} = (\mathbf{Y}_0, \ldots, \mathbf{Y}_t)$, with $\mathbf{Y}_0 = \hat{1}$ and kernel K; this trajectory is to be coupled *ex post facto* with $\vec{\mathbf{X}} = (\mathbf{X}_0, \ldots, \mathbf{X}_t)$, which is regarded as a trajectory from K. Finally, we check whether $\mathbf{Y}_t = \hat{0}$. If so, the value $\mathbf{X}_0$ is accepted as an observation from π; if not, we repeat (as for Algorithm 2.1, but always with $\mathbf{X}_t = \hat{0}$).

Remark 7.3 When $\mathcal{X}$ is finite, Algorithm 7.2 reduces to the algorithm of Section 7.2 in Fill [11].

7.1 Realizable monotonicity

It is easy to see from (6.6) that if there exists a monotone transition rule, i.e., a transition rule ϕ with the property that

$$\phi(x,u) \le \phi(y,u) \text{ for every } u \quad \text{whenever} \quad x \le y, \tag{7.1}$$

then K is stochastically monotone according to Definition 7.1(c). We call this stronger property *realizable monotonicity*. It is a common misbelief that, conversely, stochastic monotonicity for K implies realizable monotonicity. This myth is annihilated by Fill and Machida [13] and Machida [30], even for the case of a finite poset $\mathcal{X}$, for which it is shown that every stochastically monotone kernel is realizably monotone (i.e., admits a monotone transition rule) if and only if the cover graph of $\mathcal{X}$ (i.e., its Hasse diagram regarded as an undirected graph) is acyclic.

Nevertheless, realizable monotonicity can be used to motivate and explain Algorithm 7.2. Indeed, suppose for the remainder of this subsection that K admits a monotone transition rule ϕ as in (7.1). Then we proceed to build a bounding process as in Section 5.2 and show how Algorithm 2.1 can be applied efficiently. One immediately verifies by induction that

$$\mathbf{Y}_s(\hat{0}) \le \mathbf{Y}_s(x) \le \mathbf{Y}_s(\hat{1}) \quad \text{for all } 0 \le s \le t \text{ and all } x \in \mathcal{X}. \tag{7.2}$$

Thus $\mathbf{D}_s := [\mathbf{Y}_s(\hat{0}), \mathbf{Y}_s(\hat{1})] = \{y \in \mathcal{X} : \ \mathbf{Y}_s(\hat{0}) \le y \le \mathbf{Y}_s(\hat{1})\}$ gives a bounding process, and the pair $(\mathbf{Y}_s(\hat{0}), \mathbf{Y}_s(\hat{1}))$ is a quite concise representation of $\mathbf{D}_s$. In plain language, since monotonicity is preserved, when the chains $\vec{\mathbf{Y}}(\hat{0})$ and $\vec{\mathbf{Y}}(\hat{1})$ have coalesced, so must have every $\vec{\mathbf{Y}}(x)$.

But note also that, if we choose the initial state $\mathbf{X}_t$ to be $\hat{0}$, then $\{\mathbf{Y}_t(\hat{1}) = \hat{0}\}$ is the coalescence event $\mathbf{C}$. Algorithmically, it follows that if $\mathbf{X}_t = \hat{0}$ is a legitimate

starting point for Algorithm 2.1 [as discussed near the end of Section 6.2, it is sufficient for this that $\pi(\{\hat{0}\}) > 0$], and if $\mathbf{P}(\mathbf{C}) > 0$, then $\mathcal{L}(\mathbf{X}_0|\mathbf{C}) = \pi$. Informally put, we need only track the single upper-bound trajectory $\vec{\mathbf{Y}}(\hat{1})$ in the forward phase; if $\mathbf{Y}_t(\hat{1}) = \hat{0}$, then the routine (correctly) accepts $\mathbf{X}_0$ as an observation from π.

Notwithstanding the fundamental distinction between the two notions of monotonicity, the routine of Algorithm 7.2 shares with its cousin from the realizably monotone case the feature that, in the forward phase, the user need only check whether a single K-trajectory started at $\hat{1}$ ends at $\hat{0}$.

Remark 7.4 (a) Lower and upper bounding processes can also be constructed when Algorithm 2.1 is applied with a so-called "anti-monotone" transition rule; we omit the details. See Häggström and Nelander [19], Huber [23], Kendall [26], Møller [33], Møller and Schladitz [34], and Thönnes [42] for further discussion in various specialized settings. There are at least two neat tricks associated with anti-monotone rules. The first is that, by altering the natural partial order on $\mathcal{X}$, such rules can be regarded, in certain bipartite-type settings, as monotone rules, in which case the analysis of Section 7.3 (with $\hat{0}$ and $\hat{1}$ taken in the *altered* ordering, of course) is available: consult Section 3 of [19], the paper [34], and Definition 5.1 in [42]. The second is that the poset $\mathcal{X}$ is allowed to be "upwardly unbounded" and so need not have a $\hat{1}$: consult, again, [34] and [42].

(b) Dealing with monotone rules on partially ordered state spaces without $\hat{1}$ is problematic and requires the use of "dominating processes." We comment that a dominating process provides a sort of *random* bounding process and is useful when the state space is noncompact, but we shall not pursue these ideas any further here. See Kendall [26] and Kendall and Møller [27] in the context of CFTP; we hope to discuss the use of dominating processes for our algorithm in future work.

7.2 Rigorous description of Algorithm 7.2

Let K be an SM kernel with stationary distribution π and let $\widetilde{K}$ be its time-reversal, exactly as in the paragraph containing (6.5). Since we will not be using a transition rule, we otherwise forsake the development in Sections 6.2–6.3 and apply afresh the general framework of Section 6.1. For simplicity, we assume $\pi(\{\hat{0}\}) > 0$; weaker conditions are possible for sufficiently regular chains—see Machida [31].

Upward kernels: According to Theorem 1 of Kamae, Krengel, and O'Brien [25], Definition 7.1(b) is equivalent to the existence of an *upward kernel* M [i.e., a Markov kernel on $\mathcal{X}$ such that, for all x, $M(x, \cdot)$ is supported on $\{y \in \mathcal{X} : y \geq x\}$] satisfying $\nu_2 = \nu_1 M$. Thus our assumption that K is SM implies the existence of

upward kernels M_{xy}, $x \le y$, such that

$$K(y,\cdot) = \int K(x,dx')\, M_{xy}(x',\cdot) \quad \text{for all } x \le y. \tag{7.3}$$

Choose and fix such kernels M_{xy}; we will assume further that $M_{xy}(x',B)$ is jointly measurable in $(x,y,x') \in \mathcal{X}^3$ for each $B \in \mathcal{B}$.

The Markov chain and the first probability space: We consider a probability space $(\Omega, \mathcal{A}, P)$ on which are defined a Markov chain $\vec{X} = (X_0, \ldots, X_t)$ satisfying

$$P(X_0 \in dx_0, \ldots, X_t \in dx_t) = \pi(dx_0)K(x_0,dx_1)\cdots K(x_{t-1},dx_t)$$

and another process $\vec{Y} = (Y_0, \ldots, Y_t)$ such that

$$P(\vec{Y} \in d\vec{y} \,|\, \vec{X} = \vec{x}) = \delta_{\hat{1}}(dy_0)\, M_{x_0,\hat{1}}(x_1,dy_1) \cdots M_{x_{t-1},y_{t-1}}(x_t,dy_t). \tag{7.4}$$

Recalling the notational note at the end of Section 1, let

$$C := \{Y_t = \hat{0}\} \tag{7.5}$$

and observe that conditions (i)–(iv) and (v′) in Section 6.1 are satisfied by taking

$$\begin{aligned} (\mathcal{X}', \mathcal{B}') &= (\mathcal{X}, \mathcal{B}), \quad B' = \{\hat{0}\} (\in \mathcal{B}'), \quad \mathcal{B}'' = \{\emptyset, B'\}, \\ X &= X_0, \quad X' = X_t, \quad Y = Y_t, \quad C \text{ as at (7.5)}. \end{aligned} \tag{7.6}$$

Condition (v′) follows from the independence of X_0 and $\vec{Y}$, which in turn can be verified by a simple calculation using (7.4) and (7.3).

The second probability space and the algorithm: To set up the second probability space, we compute conditional probability distributions in stages, as in Section 6.2. As there,

$$P(X_0 \in dx_0, \ldots, X_{t-1} \in dx_{t-1} \,|\, X_t = x_t) = \widetilde{K}(x_t, dx_{t-1}) \cdots \widetilde{K}(x_1, dx_0)$$

serves as a conditional distribution for $(X_0, \ldots, X_{t-1})$ given $X_t = x_t$. Furthermore, (7.4) gives a conditional distribution for $\vec{Y}$ given $\vec{X}$.

We now see how to set up and simulate our second space. The user sets $\mathbf{X}_t := \hat{0}$, then draws an observation $\mathbf{X}_{t-1} = x_{t-1}$ from $\widetilde{K}(x_t,\cdot)$, then an observation $\mathbf{X}_{t-2} = x_{t-2}$ from $\widetilde{K}(x_{t-1},\cdot)$, etc. Next, having chosen $\vec{\mathbf{X}} = \vec{x}$, the user draws an observation $\vec{\mathbf{Y}} = \vec{y}$ from $\mathcal{L}(\vec{Y} \,|\, \vec{X} = \vec{x})$. [In detail, this is done by setting $\mathbf{Y}_0 := \hat{1}$, then drawing $\mathbf{Y}_1 = y_1$ from $M_{x_0,\hat{1}}(x_1,\cdot)$, then $\mathbf{Y}_2 = y_2$ from $M_{x_1,y_1}(x_2,\cdot)$, etc.] Finally, the user sets $\mathbf{C} := \{\mathbf{Y}_t = \hat{0}\}$. With the definitions (7.6), this definition of $\mathbf{C}$, and

$$\mathbf{X} = \mathbf{X}_0, \quad \mathbf{X}' = \mathbf{X}_t, \quad D(\hat{0}) = 1/\pi(\{\hat{0}\}),$$

assumptions (i)–(x) of Section 6.1 are verified in a straightforward manner. According to Corollary 6.6, if $\mathbf{P}(\mathbf{C}) > 0$, then $\mathcal{L}(\mathbf{X}_0|\mathbf{C}) = \pi$. Thus Algorithm 7.2 works as claimed.

Remark 7.5 If the stronger assumption of realizable monotonicity holds, then Algorithm 7.2 reduces to the specialization of Algorithm 2.1 discussed above. This follows from the fact that one can then take

$$M_{xy}(x', \cdot) := P(\phi(y, U_1) \in \cdot \,|\, \phi(x, U_1) = x'), \tag{7.7}$$

for the upward kernels in (7.3), provided that RHS(7.7) is jointly measurable in (x, y, x').

7.3 Performance of Algorithm 7.2

Using condition (x), we see that the routine in Algorithm 7.2 has probability

$$\mathbf{P}(\mathbf{C}) = \frac{P(C)}{\pi(\{\hat{0}\})} = \frac{K^t(\hat{1}, \{\hat{0}\})}{\pi(\{\hat{0}\})} \tag{7.8}$$

of accepting the generated value $\mathbf{X}_0$. To understand this in another way, note that our assumption $\pi(\{\hat{0}\}) > 0$ implies that $\widetilde{K}^t(\hat{0}, \cdot) \ll \pi(\cdot)$, and that the decreasing function $x \mapsto K^t(x, \{\hat{0}\})/\pi(\{\hat{0}\})$ on $\mathcal{X}$ serves as the Radon–Nikodym derivative (RND) $x \mapsto \widetilde{K}^t(\hat{0}, dx)/\pi(dx)$. With this choice of RND, we have

$$\mathbf{P}(\mathbf{C}) = \inf_x \frac{\widetilde{K}^t(\hat{0}, dx)}{\pi(dx)};$$

this last expression is a natural (though stringent) measure of agreement between the distribution $\widetilde{K}^t(\hat{0}, \cdot)$ and the stationary distribution. In the discrete case, our performance results reduce to results found in Sections 7–8 of Fill [11]; consult that paper for further discussion.

7.4 An extension: stochastic cross-monotonicity

There is no reason that the chains $\vec{\mathbf{X}}$ and $\vec{\mathbf{Y}}$ in Algorithm 7.2 need have the same kernel. Thus, consider two (possibly different) kernels K and L satisfying the *stochastic cross-monotonicity* (or *cross-SM*) property

$$K(x, \cdot) \preceq L(y, \cdot) \quad \text{whenever } x \le y;$$

if $K = L$, this reduces to the Definition 7.1(c) of SM. We can then run Algorithm 7.2, replacing "with $\mathbf{Y}_0 = \hat{1}$ and kernel K" by "with $\mathbf{Y}_0 = \hat{1}$ and kernel L." The rigorous description of this algorithm is left to the reader. The analogue of (7.3) is of course

$$L(y, \cdot) = \int K(x, dx')\, M_{xy}(x', \cdot) \quad \text{for all } x \le y; \tag{7.9}$$

and if we have *cross-monotone transition rules* ϕ_K and ϕ_L [according to the appropriate generalization of (7.1)], then one can take

$$M_{xy}(x', \cdot) := P(\phi_L(y, U_1) \in \cdot \,|\, \phi_K(x, U_1) = x'). \tag{7.10}$$

In the cross-SM case, we have the extension

$$\mathbf{P}(\mathbf{C}) = \frac{P(C)}{\pi(\{\hat{0}\})} = \frac{L^t(\hat{1}, \{\hat{0}\})}{\pi(\{\hat{0}\})}$$

of (7.8).

One application of the cross-SM case which may arise in practice (Machida plans to employ the following idea in a future application of our algorithm to the mixture problems considered by Hobert et al. [21]) is that of *stochastic dominance* [$K(x, \cdot) \preceq L(x, \cdot)$ for all $x \in \mathcal{X}$] by an SM kernel L; indeed, then $K(x, \cdot) \preceq L(x, \cdot) \preceq L(y, \cdot)$ whenever $x \leq y$. Suppose further that L has stationary distribution σ with $\sigma(\{\hat{0}\}) > 0$; then, with $\rho := \sigma(\{\hat{0}\})/\pi(\{\hat{0}\})$,

$$\mathbf{P}(\mathbf{C}) = \inf_y \frac{L^t(y, \{\hat{0}\})}{\pi(\{\hat{0}\})} = \rho \inf_y \frac{\widetilde{L}^t(\{\hat{0}\}, dy)}{\sigma(dy)}, \tag{7.11}$$

using the decreasing function $y \mapsto L^t(y, \{\hat{0}\})/\sigma(\{\hat{0}\})$ as the choice of RND $y \mapsto \widetilde{L}^t(\{\hat{0}\}, dy)/\sigma(dy)$.

The practical implication (say, for finite-state problems, as we shall assume for ease of discussion for the remainder of this section) is that if the user is unable to find a rapidly mixing SM kernel K with stationary distribution π (the distribution of interest) but can find a rapidly mixing stochastically dominant SM kernel L with stationary distribution σ "not too much larger than π" (at $\{\hat{0}\}$), then our cross-SM algorithm can be applied efficiently, provided the imputation of U_1 inherent in (7.10) can be done efficiently.

To temper enthusiasm, however, we note that the limit (namely, ρ) as $t \to \infty$ of the acceptance probability (7.11) can often be achieved in simpler fashion. Suppose, for example, that simulation from σ is easy, that $x \mapsto \pi(\{x\})$ and $x \mapsto \sigma(\{x\})$ can be computed exactly except for normalizing constants, and that $x = \hat{0}$ minimizes the ratio $\sigma(\{x\})/\pi(\{x\})$. Then one can employ elementary rejection sampling to simulate from π, with acceptance probability ρ.

8 Relation to CFTP

8.1 Comparison

How does our extension of Fill's algorithm, as given by Algorithm 2.1, compare to CFTP? As we see it, our algorithm has two main advantages and one main disadvantage.

Advantages: As discussed in Section 1 and Remark 6.9(b) and in [11], a primary advantage of our Algorithms 2.1 and 3.1 is interruptibility. A related second advantage of Algorithm 2.1 concerns memory allocation. Suppose, for example, that our state space $\mathcal{X}$ is finite and that each time-step of Algorithm 2.1, including the necessary imputation (recall Section 6.3), can be carried out using a bounded amount of memory. Then, for fixed t, our algorithm can be carried out using a fixed finite amount of memory. Unfortunately, it is rare in practice that the kernel K employed is sufficiently well analyzed that one knows in advance a value of t (and a value of the seed $\mathbf{X}_t$) giving a reasonably large probability $\mathbf{P}(\mathbf{C})$ of acceptance. Furthermore, the fixed amount of memory needed is in practice larger than the typical amount of memory allocated dynamically in a run of CFTP. See also the discussion of read-once CFTP in Section 8.2.

Disadvantage: A major disadvantage of our algorithms concerns computational complexity. We refer the reader to [11] and [12] for a more detailed discussion in the setting of realizable monotonicity (and, more generally, of stochastic monotonicity). Briefly, if no attention is paid to memory usage, our algorithms have running time competitive with CFTP: cf. Remark 6.9(c), and also the discussion in Remark 9.3(e) of [11] that the running time of our Algorithm 2.1 is, in a certain sense, best possible in the stochastically monotone setting. However, this analysis assumes that running time is measured in Markov chain steps; unfortunately, time-reversed steps can sometimes take longer than do forward steps to execute (e.g., [12]), and the imputation described in Section 6.3 is sometimes difficult to carry out. Moreover, the memory usage for naive implementation of our algorithm can be exorbitant; how to trade off speed for reduction in storage needs is described in [11].

8.2 Connection with CFTP

There is a simple connection between CFTP and our Algorithm 3.1. Indeed, suppose we carry out the usual CFTP algorithm to sample from π, using kernel K, transition rule ϕ, and driving variables $\vec{\mathbf{U}} = (\mathbf{U}_0, \mathbf{U}_{-1}, \ldots)$. Let $\mathbf{T}$ denote the backwards coalescence time and let $\mathbf{X}_0 \sim \pi$ denote the terminal state output by CFTP. Let $\mathbf{W} \sim \pi$ independent of $\vec{\mathbf{U}}$, and follow the trajectory from $\mathbf{X}_{-\mathbf{T}} := \mathbf{W}$ to $\mathbf{X}_0$; call this trajectory $\vec{\mathbf{X}} = (\mathbf{X}_{-\mathbf{T}}, \ldots, \mathbf{X}_0)$. Since $\mathbf{X}_0$ is determined solely by $\vec{\mathbf{U}}$, the random variables $\mathbf{W}$ and $\mathbf{X}_0$ are independent.

When $\hat{\pi} = \pi$ in Algorithm 3.1, the algorithm simply constructs the same probability space as for CFTP, but with the ingredients generated in a different chronological order: first $\mathbf{X}_0, \mathbf{X}_{-1}, \ldots$; then $\vec{\mathbf{U}}$ (which determines $\mathbf{T}$); then $\mathbf{W} := \mathbf{X}_{-\mathbf{T}}$. Again $\mathbf{X}_0 \sim \pi$ and $\mathbf{W} \sim \pi$ are independent.

Note that a fundamental difference between Algorithm 3.1 and CFTP is in what values they report. CFTP reports $\mathbf{X}_0$ as its observation from π, while

Algorithm 3.1 reports the value $\mathbf{W} = \mathbf{X}_{-\mathbf{T}}$.

Remark 8.1 (a) Because of this statistical independence, it does not matter in Algorithm 3.1 that we actually use $\mathbf{X}_0 \sim \hat{\pi} \neq \pi$.

(b) The fact (1) that $\mathbf{W}$, unlike $\mathbf{X}_0$, is independent of $\vec{\mathbf{U}}$, together with (2) that $\mathbf{T}$ depends solely on $\vec{\mathbf{U}}$, explains why our algorithm is interruptible and CFTP is not.

(c) In a single run of CFTP, the user would of course be unable to choose $\mathbf{W} \sim \pi$ as above, just as in a single run of Algorithm 3.1 we do not actually choose $\mathbf{X}_0 \sim \pi$. So one might regard our described connection between the two algorithms as a bit metaphorical. But see Section 9.2.

Read-once CFTP: We note also that our Algorithm 2.1 bears close resemblance to the read-once CFTP algorithm of Wilson [45]. Indeed, that algorithm shares with ours the property of requiring only a bounded amount of memory. Furthermore, in the execution of Algorithm 2.1, consider relabeling time as follows. At the k^{th} attempt at coalescence ($k = 1, 2, \ldots$), consider $\mathbf{X}_{-(k-1)t}, \ldots, \mathbf{X}_{-kt}$ in place of $\mathbf{X}_t, \ldots, \mathbf{X}_0$, respectively, where now $\mathbf{X}_{-kt}$ plays two roles: the candidate for output in attempt k and the starting state in attempt $k+1$. Then Algorithm 2.1 will output $\mathbf{X}_{-kt}$ if and only if coalescence occurs in attempt k. Read-once CFTP, on the other hand, runs chains forwards in time from time 0, outputting $\mathbf{X}_{kt}$ if and only if coalescence occurs beginning at time kt and ending at time $(k+1)t$ and has also occurred on some earlier such interval. Thus, in some sense, the two algorithms are mirrors of one another. However, the read-once algorithm is *not* interruptible, because its output (which necessarily follows a previous coalescence), conditional on rapid termination, is biased towards values resulting from fast first coalescence. By contrast, the output of Algorithm 2.1 does not require computation of a previous coalescence.

9 Perfect samples of arbitrary size

Thus far in this paper we have considered only the problem of obtaining a single observation from the distribution π of interest. What can be done to obtain a perfect sample of size n?

9.1 Elementary options

As discussed in Section 3 of Fill [11] and in Murdoch and Rosenthal [36], the simplest option is to apply, repeatedly and independently, an algorithm producing a perfect sample of size 1. When the size-1 algorithm employed is interruptible (e.g., Algorithm 2.1 or Algorithm 3.1), the resulting size-n algorithm is

both observationwise interruptible [in the sense that, for $k = 1, \ldots, n$, the kth observation output, say $\mathbf{W}_k$, and the number of Markov chain steps required, say $\mathbf{T}_k$, are conditionally independent given all randomness used in the generation of $\mathbf{W}_1, \ldots, \mathbf{W}_{k-1}$] and totally interruptible [in the sense that $\mathbf{T}_+ := \sum_{k=1}^{n} \mathbf{T}_k$ and $\vec{\mathbf{W}} := (\mathbf{W}_1, \ldots, \mathbf{W}_n)$ are independent]. A related observation is that the conditional distribution of a fixed-duration sample given its size is that of an i.i.d. sample, again provided that time is measured in Markov chain steps; contrast comment 3 concerning CFTP in Remark 5.3 of [11].

Another simple option is to generate a single observation, say $\mathbf{W}_1$, from a size-1 perfect sampler and then run the chain K (or $\widetilde{K}$, or any other chain with stationary distribution π) for $n-1$ steps from $\mathbf{W}_1$, obtaining $\vec{\mathbf{W}} = (\mathbf{W}_1, \ldots, \mathbf{W}_n)$. Note that each observation $\mathbf{W}_k$ is marginally distributed as π. This size-n algorithm is also observationwise interruptible and totally interruptible. Of course, statistical use of $\vec{\mathbf{W}}$ is complicated by the serial dependence of its entries.

A third option is to compromise between the first two ideas and generate ν independent vectors $\vec{\mathbf{W}}^{(1)}, \ldots, \vec{\mathbf{W}}^{(\nu)}$, where $\vec{\mathbf{W}}^{(i)}$ is obtained using the size-t_i sampler described in the preceding paragraph and ν and $(t_1, \ldots, t_\nu)$ are chosen (in advance, deterministically) so that $t_1 + \cdots + t_\nu = n$. Again we have both forms of interruptibility. This third option is an interruptible analogue of the "RCFTP (Repeated CFTP) tours" of Murdoch and Rosenthal [36].

9.2 Efficient use of size-1 perfect samplers

All three of the options in Section 9.1 seem wasteful. After all, a great deal of randomness and computational effort goes into the use of Algorithm 2.1 or 3.1, yet only a single observation from π comes out. Is there a way to be more efficient? Our short answer is this: Yes, but only for Algorithm 3.1, and then the advantage of interruptibility is forfeited. The remainder of this subsection provides an explanation.

Using Algorithm 2.1*:* Suppose that we have in hand an observation $\mathbf{W}_1 \sim \pi$, for example from a first run of Algorithm 2.1. If we feed $\mathbf{W}_1$ into the routine of Algorithm 2.1 as $\mathbf{X}_t$, then the resulting $\vec{\mathbf{X}}$ is unconditionally distributed as a stationary trajectory from K; in particular, $\mathbf{X}_s \sim \pi$ for $0 \le s \le t$. Unfortunately, this is not generally true conditionally given success (i.e., given coalescence):

Example 9.1 Consider again the toy random-walk example of Section 4, again with $t = 2$ but now with the modifications

$$k(0,0) = k(2,2) = 3/4, \quad k(0,1) = k(2,1) = 1/4, \quad k(1,0) = k(1,2) = 1/2,$$
$$k(0,2) = k(1,1) = k(2,0) = 0$$

and $\pi = (2/5, 1/5, 2/5)$ to K and π. Suppose that the seed value $\mathbf{X}_2$ is chosen according to π, and that the independent-transitions rule is used. Then of course

$\mathcal{L}(\mathbf{X}_0|\mathbf{C}) = \pi$, but one can check that $\mathcal{L}(\mathbf{X}_1|\mathbf{C}) = (7/22, 8/22, 7/22)$ and that $\mathcal{L}(\mathbf{X}_2|\mathbf{C}) = (4/11, 3/11, 4/11)$.

This example illustrates a rather catastrophic fact about Algorithm 2.1: If we use the output $\mathbf{W}_2 := \mathbf{X}_0$ as an observation from π, we may no longer use $\mathbf{W}_1 = \mathbf{X}_t$ as such. We know of no systematic way to make use of the auxiliary randomness generated in a run of Algorithm 2.1.

Using Algorithm 3.1*:* Let Algorithm 3.1′ denote Algorithm 3.1 modified as follows. For fixed t_0, use $\mathbf{T}' := \max(\mathbf{T}, t_0 - 1)$ in place of $\mathbf{T}$. Algorithm 3.1′ uses a conservative detection rule (see section 5), and so is still valid.

Suppose again that we have in hand an observation distributed as π, say from a first run of Algorithm 3.1′ or 3.1 or 2.1. If we call this observation $\mathbf{W}_{t_0}^{(1)}$ and feed it into Algorithm 3.1′ as $\mathbf{X}_0$, then the resulting trajectory $(\ldots, \mathbf{X}_{-2}, \mathbf{X}_{-1}, \mathbf{X}_0)$ is distributed as a stationary trajectory from K; in particular,

$$\vec{\mathbf{W}}^{(1)} = (\mathbf{W}_1^{(1)}, \ldots, \mathbf{W}_{t_0}^{(1)}) := (\mathbf{X}_{-(t_0-1)}, \ldots, \mathbf{X}_0)$$

is a stationary trajectory of length t_0 from K. Defining $\mathbf{W}_{t_0}^{(2)} := \mathbf{X}_{-\mathbf{T}}$, we now feed $\mathbf{W}_{t_0}^{(2)}$ into Algorithm 3.1′ as the new seed $\mathbf{X}_0$; using randomness otherwise independent of the first run, we obtain another stationary trajectory, call it $\vec{\mathbf{W}}^{(2)}$, of length t_0 from K. Taking $\mathbf{W}_{t_0}^{(3)}$ to be the value of $\mathbf{X}_{-\mathbf{T}}$ from this second run, we then provide $\mathbf{W}_{t_0}^{(3)}$ as a seed producing $\vec{\mathbf{W}}^{(3)}$, and so on.

It is not hard to see that, for any $\nu \geq 1$, the joint distribution of the "tours" (as Murdoch and Rosenthal [36] call them) $\vec{\mathbf{W}}^{(\nu)}, \ldots, \vec{\mathbf{W}}^{(2)}, \vec{\mathbf{W}}^{(1)}$ is the same as the joint distribution of ν serially generated "Guarantee Time CFTP" (GTCFTP) tours as in Section 5 of [36]. (Their "guarantee time" T_g is our $t_0 - 1$.) Since each tour $\vec{\mathbf{W}}^{(i)}$ has the marginal distribution

$$\mathbf{P}(\vec{\mathbf{W}}^{(i)} \in d\vec{w}) = \pi(dw_1)\, K(w_1, dw_2) \cdots K(w_{t_0-1}, dw_{t_0}) \tag{9.1}$$

and since both $(\vec{\mathbf{W}}^{(1)}, \vec{\mathbf{W}}^{(2)}, \ldots, \vec{\mathbf{W}}^{(\nu)})$ and the sequence of GTCFTP tours are clearly tour-valued Markov chains, we see that our tour-chain is simply the stationary time-reversal of the (stationary) GTCFTP tour-chain, where the common stationary tour-distribution is (9.1).

Since GTCFTP tours are analyzed by Murdoch and Rosenthal [36], we refer the interested reader to [36] for further discussion and analysis. One highlight is that while the tours $\vec{\mathbf{W}}^{(1)}, \vec{\mathbf{W}}^{(2)}, \ldots$ are not independent, they are 1-dependent: that is, $\vec{\mathbf{W}}^{(i)}$ and $(\vec{\mathbf{W}}^{(1)}, \vec{\mathbf{W}}^{(2)}, \ldots, \vec{\mathbf{W}}^{(i-2)}, \vec{\mathbf{W}}^{(i+2)}, \vec{\mathbf{W}}^{(i+3)}, \ldots)$ are independent for every i.

Remark 9.2 For $t_0 = 1$, one can check that the tour-algorithm we have described is an observationwise interruptible and totally interruptible algorithm for producing

independent observations from π. Unfortunately, for $t_0 \geq 2$, one can check that all interruptibility is lost, due to the dependence of $(\mathbf{X}_{-(t_0-1)}, \ldots, \mathbf{X}_{-1})$ and $\mathbf{X}_{-T}$ (unlike the independence of $\mathbf{X}_0$ and $\mathbf{X}_{-T}$) in Algorithm 3.1′.

In light of the above remark, and the fact that CFTP tours are generated considerably more easily than are our Algorithm 3.1′ tours, we see nothing to recommend the use of Algorithm 3.1′ in practice. (We remark, nonetheless, that it was in investigating our tours that we discovered the connection between Algorithm 3.1 and CFTP described in Section 8.2.)

Acknowledgments. We thank the anonymous referees for many helpful suggestions about the content and exposition of this paper, including the connections between Algorithm 2.1 and read-once CFTP. We also thank David Wilson for helpful comments.

References

[1] Bidigare, P., Hanlon, P., and Rockmore, D. (1999). A combinatorial description of the spectrum for the Tsetlin library and its generalization to hyperplane arrangements. *Duke Mathematics Journal* **99** 135–174.

[2] Brooks, S. P., Dellaportas, P., and Roberts, G. O. (1997). An approach to diagnosing total variation convergence of MCMC algorithms. *Journal of Computational and Graphical Statistics* **6** 251–265.

[3] Brown, K. and Diaconis, P. (1998). Random walk and hyperplane arrangements. *Annals of Probability* **26** 1813–1854.

[4] Corcoran, J. and Tweedie, R. L. (1999). Perfect simulation of Harris recurrent Markov chains. Preprint, Colorado State University.

[5] Cowles, M. K. and Carlin, B. P. (1996). Markov chain Monte Carlo convergence diagnostics: a comparative review. *Journal of the American Statistical Association* **91** 883–904.

[6] Devroye, L. (1986). Nonuniform random variate generation. Springer–Verlag, New York.

[7] Diaconis, P. and Fill, J. A. (1990). Strong stationary times via a new form of duality. *Annals of Probability* **18** 1483–1522.

[8] Diaconis, P. and Freedman, D. (1999). Iterated random functions. *SIAM Review* **41** 45–76.

[9] Dobrow, R. P. and Fill, J. A. (1995). On the Markov chain for the move-to-root rule for binary search trees. *Annals of Applied Probability* **5** 1–19.

[10] Dobrow, R. P. and Fill, J. A. (1995). Rates of convergence for the move-to-root Markov chain for binary search trees. *Annals of Applied Probability* **5** 20–36.

[11] Fill, J. A. (1998). An interruptible algorithm for perfect sampling via Markov chains. *Annals of Applied Probability* **8** 131–162.

[12] Fill, J. A. (1998). The move-to-front rule: a case study for two perfect sampling algorithms. *Probability in the Engineering and Informational Sciences* **12** 283–302.

[13] Fill, J. A. and Machida, M. (1998). Stochastic and realizable monotonicity. Preprint. Available from `http://www.mts.jhu.edu/~fill/`.

[14] Fill, J. A., Machida, M., Murdoch, D., and Rosenthal, J. (2000). Extension of Fill's perfect rejection sampling algorithm to general chains (extended abstract). Pages 37–52 in *Monte Carlo Methods* (ed.: N. Madras), *Fields Institute Communications* **26**, American Mathematical Society.

[15] Foss, S. G. and Tweedie, R. L. (1998). Perfect simulation and backward coupling. *Stochastic Models* **14** 187–203.

[16] Gelfand, A. E. and Smith, A. F. M. (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association* **85** 398–409.

[17] Gilks, W. R., Richardson, S., and Spiegelhalter, D. J., editors (1996). *Markov Chain Monte Carlo in Practice.* Chapman and Hall.

[18] Green, P. J. and Murdoch, D. J. (1999). Exact sampling for Bayesian inference: towards general purpose algorithms. *Bayesian statistics, 6 (Alcoceber, 1998)*, 301–321, Oxford Univ. Press, New York.

[19] Häggström, O. and Nelander, K. (1998). Exact sampling from anti-monotone systems. *Statistica Neerlandica* **52** 360–380.

[20] Häggström, O. and Nelander, K. (1999). On exact simulation of Markov random fields using coupling from the past. *Scandinavian Journal of Statistics* **26(3)** 395–411.

[21] Hobert, J. P., Robert, C. P., and Titterington, D. M. (1999). On perfect simulation for some mixtures of distributions. *Statistics and Computing* **9** 287–298.

[22] Huber, M. (1998). Efficient exact sampling from the Ising model using Swendsen–Wang. A two-page version appears in *Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*. Preprint.

[23] Huber, M. (1998). Exact sampling and approximate counting techniques. In *Proceedings of the 30th ACM Symposium on the Theory of Computing*, 31–40.

[24] Huber, M. (1998). Interruptible exact sampling and construction of strong stationary times for Markov chains. Preprint.

[25] Kamae, T., Krengel, U., and O'Brien, G. L. (1977). Stochastic inequalities on partially ordered state spaces. *Ann. Probab.* **5** 899–912.

[26] Kendall, W. (1998). Perfect simulation for the area-interaction point process. In Accardi, L. and Heyde, C. C., editors, *Probability Towards 2000* 218–234. Springer.

[27] Kendall, W. and Møller, J. (1999). Perfect Metropolis–Hastings simulation of locally stable point processes. *Advances in Applied Probability*, to appear.

[28] Kendall, W. and Thönnes, E. (1999). Perfect simulation in stochastic geometry. *Pattern Recognition* **32** 1569–1586. Special issue on random sets.

[29] Kifer, Y. (1986). *Ergodic Theory of Random Transformations.* Birkhäuser, Boston.

[30] Machida, M. (1999). Stochastic monotonicity and realizable monotonicity. Ph.D. dissertation, Department of Mathematical Sciences, The Johns Hopkins University. Available from `http://www.mts.jhu.edu/~machida/`.

[31] Machida, M. (2000). The FMMR algorithm for a stochastically monotone kernel in the absolutely continuous case. Draft available from `http://www.mts.jhu.edu/~machida/`.

[32] Meyn, S. P. and Tweedie, R. L. (1994). Computable bounds for convergence rates of Markov chains. *Annals of Applied Probability* **4** 981–1011.

[33] Møller, J. (1999). Perfect simulation of conditionally specified models. *Journal of the Royal Statistical Society, Series B* **61** 251–264.

[34] Møller, J. and Schladitz, K. (1999). Extensions of Fill's algorithm for perfect simulation. *Journal of the Royal Statistical Society, Series B* **61** 955–969.

[35] Murdoch, D. J. and Green, P. J. (1998). Exact sampling from a continuous state space. *Scandinavian Journal of Statistics* **25** 483–502.

[36] Murdoch, D. J. and Rosenthal, J. S. (1998). Efficient use of exact samples. *Statistics and Computing*, to appear.

[37] Propp, J. G. and Wilson, D. B. (1996). Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms* **9** 223–252.

[38] Propp, J. G. and Wilson, D. B. (1998). Coupling from the past: a user's guide. In Aldous, D. and Propp, J. G., editors, *Microsurveys in Discrete Probability,* volume 41 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* 181–192. American Mathematical Society.

[39] Propp, J. G. and Wilson, D. B. (1998). How to get a perfectly random sample from a generic Markov chain and generate a random spanning tree of a directed graph. *Journal of Algorithms* **27** 170–217.

[40] Rosenthal, J. S. (1995). Minorization conditions and convergence rates for Markov chain Monte Carlo. *Journal of the American Statistical Association* **90** 558–566.

[41] Smith, A. F. M. and Roberts, G. O. (1993). Bayesian computation via the Gibbs sampler and related Markov chain Monte Carlo methods (with discussion). *Journal of the Royal Statistical Society, Series B* **55** 3–23.

[42] Thönnes, E. (1999). Perfect simulation of some point processes for the impatient user. *Advances in Applied Probability* **31** 69–87.

[43] Tierney, L. (1994). Markov chains for exploring posterior distributions (with discussion). *Annals of Statistics* **22** 1701–1762.

[44] Wilson, D. B. (1998). Annotated bibliography of perfectly random sampling with Markov chains. In Aldous, D. and Propp, J., editors, *Microsurveys in Discrete Probability*, volume 41 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* 209–220. American Mathematical Society. Latest updated version is posted at `http://dimacs.rutgers.edu/~dbwilson/exact/`.

[45] Wilson, D. B. (2000). How to couple from the past using a read-once source of randomness. *Random Structures and Algorithms* **16** 85–113.